**TÌM HIỂU NGÔN NGỮ LẬP TRÌNH PYTHON**

**Mục lục**

[**I.** **INFORMATION - THÔNG TIN** 2](#_Toc118752290)

[**II.** **VARIABLE - BIẾN** 3](#_Toc118752291)

[**1.** **Phép gán** 3](#_Toc118752292)

[**a.** **Kiểu dữ liệu chuẩn:** 3](#_Toc118752293)

[**b.** **Gán giá trị cho biến:** 3](#_Toc118752294)

[**c.** **Gán đồng thời cho nhiều biến:** 4](#_Toc118752295)

[**2.** **Numberic types** 4](#_Toc118752296)

[**3.** **Text types** 5](#_Toc118752297)

[**4.** **Sequence types** 6](#_Toc118752298)

[**5.** **Dictionary** 9](#_Toc118752299)

[**6.** **Set** 9](#_Toc118752300)

[**7.** **Boolean** 10](#_Toc118752301)

[**III.** **CÚ PHÁP** 10](#_Toc118752302)

[**1.** **Thụt lề** 10](#_Toc118752303)

[**2.** **Toán tử cơ bản** 11](#_Toc118752304)

[**a.** **Toán tử số học** 11](#_Toc118752305)

[**b.** **Toán tử quan hệ** 11](#_Toc118752306)

[**c.** **Toán tử gán** 11](#_Toc118752307)

[**d.** **Toán tử logic** 12](#_Toc118752308)

[**e.** **Toán tử biwter** 12](#_Toc118752309)

[**f.** **Toán tử khai thác** 12](#_Toc118752310)

[**g.** **Toán tử xác thực** 12](#_Toc118752311)

[**3.** **Câu lệnh và luồng điều khiển** 13](#_Toc118752312)

1. **INFORMATION - THÔNG TIN**

* Python là một ngôn ngữ lập trình phổ biến. Nó được tạo ra bởi Guido Van Rossum, và được phát hành vào năm 1991.
* Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.
* Python được phát triển để chạy trên nền  đến [Windows](https://vi.wikipedia.org/wiki/Microsoft_Windows), [Mac OS](https://vi.wikipedia.org/wiki/Mac_OS), [Linux](https://vi.wikipedia.org/wiki/Linux) và [nhiều hệ điều hành khác thuộc họ Unix](https://vi.wikipedia.org/wiki/T%C6%B0%C6%A1ng_t%E1%BB%B1_Unix).
* Python hoàn toàn [tạo kiểu động](https://vi.wikipedia.org/w/index.php?title=T%E1%BA%A1o_ki%E1%BB%83u_%C4%91%E1%BB%99ng&action=edit&redlink=1) và dùng cơ chế [cấp phát bộ nhớ tự động](https://vi.wikipedia.org/wiki/Qu%E1%BA%A3n_l%C3%BD_b%E1%BB%99_nh%E1%BB%9B); do vậy nó tương tự như [Perl](https://vi.wikipedia.org/wiki/Perl), [Ruby](https://vi.wikipedia.org/wiki/Ruby_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), [Scheme](https://vi.wikipedia.org/wiki/Scheme), [Smalltalk](https://vi.wikipedia.org/wiki/Smalltalk), và [Tcl](https://vi.wikipedia.org/wiki/Tcl).
* Python có thể làm gì?
  + Python có thể được sử dụng trên máy chủ để tạo các ứng dụng web.
  + Python có thể được sử dụng cùng với phần mềm để tạo quy trình công việc.
  + Python có thể kết nối với các hệ thống cơ sở dữ liệu. Nó cũng có thể đọc và sửa đổi các tập tin.
  + Python có thể được sử dụng để xử lý dữ liệu lớn và thực hiện toán học phức tạp.
  + Python có thể được sử dụng để tạo mẫu nhanh, hoặc để phát triển phần mềm sẵn sàng sản xuất.
* Nó thường được sử dụng cho:
  + Lập trình web
  + Lập trình phần mềm
  + Công cụ toán học
  + Kịch bản hệ thống

1. **VARIABLE - BIẾN**
2. **Phép gán**
   1. **Kiểu dữ liệu chuẩn:**

Dữ liệu được lưu trữ trong bộ nhớ có thể có nhiều loại. Mỗi biến sau khi được khai báo sẽ mang một kiểu dữ liệu. Với mỗi kiểu dữ liệu sẽ có các thao tác có thể thực hiện và phương thức lưu trữ cho từng loại.

Python có 6 loại dữ liệu phổ biến:

* + **Numberic types**
  + **Text types**
  + **Sequence types**
  + **Mapping types - Dictionary**
  + **Set types**
  + **Boolean types**
  1. **Gán giá trị cho biến:**

Không cần khai báo tên biến cho phép gán. Chương trình tự động khai báo khi bạn gán giá trị cho một biến. Để thực hiện phép gán ta sử dụng dấu bằng (“**=**”) theo cú pháp:

toán hạng 1 **=** toán hạng 2

Với toán hạng 1 chính là tên biến và toán hạng 2 chính là giá trị được lưu trong biến.

Ví dụ:

|  |  |
| --- | --- |
| **age = 18**  **height = 1.7**  **name = “Phuc”** | **# số nguyên**  **# số thực**  **# xâu, chuỗi** |

* 1. **Gán đồng thời cho nhiều biến:**

Python cho phép bạn gán một giá trị duy nhất cho đồng thời nhiều biến khác nhau:

biến 1 **=** biến 2 **=** … **=** giá trị

Ví dụ:

|  |  |
| --- | --- |
| **a = b = c = 18** | **Đồng nghĩa với:**  **a = 18**  **b = 18**  **c = 18** |

Ngoài ra, bạn cũng có thể đồng thời gán nhiều giá trị cho nhiều biến:

biến 1, biến 2, … **=** giá trị 1, giá trị 2, …

Ví dụ:

|  |  |
| --- | --- |
| **a, b, c = 18, 1.7, “Phuc”** | **Đồng nghĩa với:**  **a = 18**  **b = 1.7**  **c = “Phuc”** |

1. **Numberic types**

Python hỗ trợ 4 loại số:

* + **int** - số nguyên
  + **long** - số nguyên dài
  + **float** - số thực
  + **complex** - số phức

**Lưu ý:**

* + Python cho phép bạn sử dụng **l** hoặc **L** cho kiểu dữ liệu **long**. Chương trình sẽ hiển thị số nguyên dài có chữ **L** in hoa.
  + Một số phức được kí hiệu **x + yj**, trong đó **x**, **y** là số thực và **j** là đơn vị ảo.

**phiên bản**

2.7 về trước

3.0 tới giờ

**int**

32 bit

vô hạn

**long**

vô hạn

vô hạn

Ví dụ:

|  |  |
| --- | --- |
| **a = 18**  **b = 1L**  **c = 1.7**  **d = 4+2j** | **# int**  **# long**  **# float**  **# complex** |

1. **Text types**

Chuỗi trong Python là tập hợp các ký tự liền kề được biểu thị trong dấu nháy đơn hoặc nháy kép.

|  |  |
| --- | --- |
| **str[i]** | trả về ký tự thứ **i+1** của chuỗi **str** |
| **str[i : j]** | trả về chuỗi các ký tự từ vị trí thứ **i+1** tới vị trí thứ **j** trong chuỗi **str** |
| **str[i :]** | trả về chuỗi các ký tự từ vị trí thứ **i+1** tới hết chuỗi **str** |
| **str \* x** | trả về chuỗi **str** lặp lại **x** lần |
| **str + wrd** | trả về chuỗi bằng cách nối hai chuỗi **str** và **wrd** lại |

Ví dụ:

|  |  |
| --- | --- |
| **str, wrd = “Hello”, “World!”** | |
| **str**  **str[1]**  **str[2 : 4]**  **str[1 :]**  **str \* 2**  **str + wrd** | **Hello**  **e**  **ll**  **ello**  **HelloHello**  **HelloWorld!** |

1. **Sequence types**

Python hỗ trợ 3 loại dãy:

* + **list** - danh sách
  + **tuple** - bộ dữ liệu
  + **range** - đoạn

**list**

Danh sách là kiểu dữ liệu linh hoạt nhất trong các kiểu dữ liệu hỗn hợp của Python. Danh sách chứa các phần tử được tách bằng dấu phẩy (“**,**”) và được đặt trong dấu ngoặc vuông (“**[**”, “**]**”).

Danh sách có thể tương tự như mảng ở các ngôn ngữ lập trình khác, tuy nhiên khác biệt giữa chúng là danh sách có thể lưu nhiều phần tử thuộc nhiều kiểu dữ liệu khác nhau.

Giá trị các phần tử cũng như kích thước danh sách có thể thay đổi được.

|  |  |
| --- | --- |
| **lst[i]** | trả về phần tử thứ **i+1** của danh sách **lst** |
| **lst[i : j]** | trả về các phần tử từ vị trí thứ **i+1** tới vị trí thứ **j** trong danh sách **lst** |
| **lst[i :]** | trả về các phần tử từ vị trí thứ **i+1** tới hết danh sách **lst** |
| **lst \* x** | trả về danh sách **lst** lặp lại **x** lần |
| **lst + ds** | trả về danh sách bằng cách nối hai danh sách **lst** và **ds** lại |

Ví dụ:

|  |  |
| --- | --- |
| **lst = [18, 1.7, ‘Phuc’]**  **ds = [1, 0.4, ‘love’]** | |
| **lst**  **lts[0]**  **ds[1 : 3]**  **lst[1 :]**  **ds \* 2**  **lst + ds** | **[18, 1.7, ‘Phuc’]**  **18**  **[0.4, ‘love’]**  **[1.7, ‘Phuc’]**  **[1, 0.4, ‘love’, 1, 0.4, ‘love’]**  **[18, 1.7, ‘Phuc’, 1, 0.4, ‘love’]** |

**tuple**

Bộ dữ liệu cũng là bộ kiểu dữ liệu hỗn hợp như danh sách. Tuy nhiên bộ dữ liệu được đặt trong dấu ngoặc đơn (“**(**”, “**)**”) và giá trị các phần tử cũng như kích thước của bộ dữ liệu thì không thể thay đổi được. Bộ dữ liệu có thể được coi là danh sách chỉ dùng để đọc.

|  |  |
| --- | --- |
| **tup[i]** | trả về phần tử thứ **i+1** của bộ dữ liệu **tup** |
| **tup[i : j]** | trả về các phần tử từ vị trí thứ **i+1** tới vị trí thứ **j** trong bộ dữ liệu **tup** |
| **tup [i :]** | trả về các phần tử từ vị trí thứ **i+1** tới hết bộ dữ liệu **tup** |
| **tup \* x** | trả về bộ dữ liệu **tup** lặp lại **x** lần |
| **tup + bdl** | trả về bộ dữ liệu bằng cách nối hai bộ dữ liệu **tup** và **bdl** lại |

Ví dụ:

|  |  |
| --- | --- |
| **lst = (18, 1.7, ‘Phuc’)**  **ds = (1, 0.4, ‘love’)** | |
| **tup**  **tup[0]**  **bdl[1 : 3]**  **tup[1 :]**  **bdl \* 2**  **tup+ bdl** | **(18, 1.7, ‘Phuc’)**  **18**  **(0.4, ‘love’)**  **(1.7, ‘Phuc’)**  **(1, 0.4, ‘love’, 1, 0.4, ‘love’)**  **(18, 1.7, ‘Phuc’, 1, 0.4, ‘love’)** |

**range**

ranger là danh sách các số nguyên.

|  |  |
| --- | --- |
| **range(x)** | trả về danh sách các số từ **0** đến **x-1** |
| **range(x, y)** | trả về danh sách các số từ **x** đến **y-1** |
| **range(x, y, z)** | trả về danh sách các số từ **x** đến **y-1** với bước nhảy là **z** |

Ví dụ:

|  |  |
| --- | --- |
| **range(4)**  **range(2, 6)**  **range(2, 8, 3)** | **[0, 1, 2, 3]**  **[2, 3, 4, 5]**  **[2, 5]** |

1. **Dictionary**

Từ điển là loại bảng băm bao gồm các cặp từ khóa – giá trị. Từ điển được bao quanh bởi dấu ngoặc nhọn (“**{**”, “**}**”), các giá trị có thể được chỉ định và truy cập bằng dấu ngoặc vuông (“**[**”, “**]**”).

|  |  |
| --- | --- |
| **dic** | Trả về từ điển **dic** |
| **dic[x]** | trả về các giá trị có từ khóa là **x** trong từ điển **dic** |
| **dic[a] = b** | Thêm từ khóa **a** có giá trị là **b** vào từ điển **dic** hoặc gán giá trị **b** vào từ khóa **a** đã có sẵn trong **dic** |
| **dic.keys()** | trả về danh sách chứa tất cả các từ khóa của từ điển **dic** |
| **dic.values()** | trả về danh sách chứa tất cả các giá trị của từ điển **dic** |

Ví dụ:

|  |  |
| --- | --- |
| **dic = {‘age’ : 18, ‘height’ : 1.7}** | |
| **dic[‘age’]**  **dic[‘name’] = ‘phuc’**  **dic.keys()**  **dic.values()** | **18**  **[‘age’, ‘height’, ‘name’]**  **[18, 1.7, ‘phuc’]** |

1. **Set**

Set được sử dụng để lưu trữ nhiều mục trong một biến duy nhất. Các phần tử trong set đều đôi một khác nhau.

Ví dụ:

|  |
| --- |
| **dic = {18, 1.7. ‘phuc’}** |

1. **Boolean**

**Booleans** đại diện cho một trong hai giá trị: **true** (đúng) hoặc **false** (sai). Các phép so sách sẽ trả về giá trị **boolean**.

**Boolean** có thể thay thế bởi một số

* + - 0 là **fasle**
    - Còn lại là **true**

Ví dụ:

|  |  |
| --- | --- |
| **1 > 3**  **2 < 4**  **1**  **0** | **false**  **true**  **true**  **false** |

1. **CÚ PHÁP**
2. **Thụt lề**

Python sử dụng thụt lề bằng [khoảng trắng](https://vi.wikipedia.org/w/index.php?title=K%C3%AD_t%E1%BB%B1_kho%E1%BA%A3ng_tr%E1%BA%AFng&action=edit&redlink=1) hoặc ký tự tab thay vì dùng ngoặc nhọn như trong C/C++ hay các từ khoá như trong các ngôn ngữ lập trình khác để giới hạn [khối lệnh](https://vi.wikipedia.org/wiki/Kh%E1%BB%91i_(l%E1%BA%ADp_tr%C3%ACnh)).

Lề thường được thụt vào từ câu lệnh bắt đầu và thụt ra sau câu lệnh kết thúc của khối lệnh hiện tại. Vì thế chương trình sẽ có thể thực hiện một cách chính xác cấu trúc của chương trình đó.

Tính năng này được gọi là "[quy tắc việt vị](https://vi.wikipedia.org/w/index.php?title=Quy_t%E1%BA%AFc_vi%E1%BB%87t_v%E1%BB%8B&action=edit&redlink=1)", đôi khi xuất hiện ở một số ngôn ngữ, tuy nhiên trong phần lớn ngôn ngữ thì thụt lề không phụ thuộc vào cú pháp.

Cỡ thụt lề được khuyến cáo là một phím tab = bốn dấu cách.

1. **Toán tử cơ bản**
   1. **Toán tử số học**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **+** | cộng | **a + b** |
| **-** | trừ | **a – b** |
| **\*** | nhân | **a \* b** |
| **/** | chia | **a / b** |
| **%** | chia lấy dư | **a % b** |
| **//** | chia lấy nguyên | **a // b** |
| **\*\*** | lũy thừa | **a \*\* b** |

* 1. **Toán tử quan hệ**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **<** | so sánh bé hơn | **a < b** |
| **<=** | so sánh bé hơn hoặc bằng | **a <= b** |
| **>** | so sánh lớn hơn | **a > b** |
| **>=** | so sánh lớn hơn hoặc bằng | **a >= b** |
| **==** | so sánh bằng | **a == b** |
| **!=** | so sánh khác | **a != b** |

* 1. **Toán tử gán**

|  |  |  |  |
| --- | --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** | **Đồng nghĩa** |
| **=** | gán bằng | **a = b** |  |
| **+=** | gán cộng bằng | **a += b** | **a = a + b** |
| **-=** | gán trừ bằng | **a -= b** | **a = a - b** |
| **\*=** | gán nhân bằng | **a \*= b** | **a = a \* b** |
| **/=** | gán chia bằng | **a /= b** | **a = a / b** |
| **//=** | gán chia lấy dư bằng | **a //= b** | **a = a // b** |
| **\*\*=** | gán lũy thừa bằng | **a \*\*= b** | **a = a \*\* b** |

* 1. **Toán tử logic**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **and** | và | **a and b** |
| **or** | hoặc | **a or b** |
| **not** | phủ định | **not a** |

* 1. **Toán tử biwter**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **&** | phép and | **a & b** |
| **|** | phép or | **a | b** |
| **^** | phép xor | **a ^ b** |
| **~** | phép not | **~ a** |
| **<<** | phép dời trái | **a << b** |
| **>>** | phép dời phải | **a >> b** |

* 1. **Toán tử khai thác**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **in** | kiểm tra thuộc tập | **a in b** |
| **not in** | kiểm tra không thuộc tập | **a not in b** |

* 1. **Toán tử xác thực**

|  |  |  |
| --- | --- | --- |
| **Kí hiệu** | **Công dụng** | **Source code** |
| **is** | kiểm tra bằng | **a is b** |
| **not is** | kiểm tra khác | **a not is b** |

1. **Câu lệnh và luồng điều khiển**

|  |  |  |
| --- | --- | --- |
| **Lệnh** | **Công dụng** | **Source code** |
| if | thực thi một khối lệnh nếu thoả mãn điều kiện. | **if 5 > 3:**  **print('true')** |
| else  (đi chung với if) | thực thi một khối lệnh nếu không thoả mãn điều kiện. | **if a % 2 == 0:**  **print('chan')**  **else:**  **print('le');** |
| for | lặp qua một đối tượng lặp được, gán mỗi phần tử và một biến cục bộ để sử dụng trong khối lệnh của vòng lặp. | **for x in range(7):**  **print(x)** |
| while | thực thi một khối lệnh chừng nào điều kiện còn đúng. | **while a > b:**  **a = a + 1** |
| raise | được dùng để nâng một ngoại lệ hoặc nâng lại một ngoại lệ đã được bắt từ trước. | **if a != b:**  **raise Exception('a must be equal to b')** |
| class | thực thi một khối lệnh và gắn không gian tên cục bộ của nó vào một [lớp](https://vi.wikipedia.org/wiki/L%E1%BB%9Bp_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)), để dùng trong lập trình hướng đối tượng. | **class mc:**  **name = 'phuc'**  **age = 18** |
| define | định nghĩa một [hàm](https://vi.wikipedia.org/wiki/Ch%C6%B0%C6%A1ng_tr%C3%ACnh_con) hoặc [phương thức](https://vi.wikipedia.org/wiki/Ph%C6%B0%C6%A1ng_th%E1%BB%A9c_(l%E1%BA%ADp_tr%C3%ACnh_m%C3%A1y_t%C3%ADnh)). | **def maximum(a, b):**  **if a > b:**  **return a**  **return b** |
| with | bao bọc một khối lệnh bằng một bộ quản lí ngữ cảnh (context manager) (ví dụ như khoá luồng lại trước khi chạy mã rồi mở khoá, hoặc mở một [tệp](https://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin) rồi đóng tệp lại), cho phép các hành vi kiểu RAII (sự đạt được tài nguyên là sự khởi tạo) và thay thể cho các câu lệnh try/finally thường thấy. | **with open('name', 'w') as file:**  **file.write('Phuc')** |
| break | thoát ra khỏi vòng lặp. | **while a > b:**  **a = a + 1**  **if a == 10:**  **break** |
| continue | bỏ qua lần lặp này và tiếp tục với mục kế tiếp. | **while a > b:**  **a = a + 1**  **if a % 2 == 0:**  **continue** |
| del | loại bỏ một biến, tức là tham chiếu từ tên đến giá trị sẽ bị xoá và cố gắng sử dụng biến đó sẽ gây lỗi. Một biến đã bị xoá có thể được gán lại. | **del name**  **del a, b, c** |
| pass | đóng vai trò như là một dạng [NOP](https://vi.wikipedia.org/w/index.php?title=NOP_(m%C3%A3)&action=edit&redlink=1). Câu lệnh này được dùng để tạo các khối lệnh rỗng. | **if x >= 5:**  **pass**  **else:**  **print('try again')** |
| assert | được dùng trong khi gỡ lỗi để kiểm tra điều kiện nên đúng. | **assert x >= 5**  **print('you are pass')** |
| yield | trả lại giá trị từ một hàm bộ sinh; bản thân yield cũng là một toán tử. Dạng này được dùng để triển khai các [đồng thường trình](https://vi.wikipedia.org/wiki/Coroutine). | **def sqr(a):**  **for i in a:**  **yield i \* i**  **arr = sqr(range(6))**  **for i in arr:**  **print(i)** |
| return | trả lại một giá trị từ một hàm hay phương thức. | **def maximum(a, b):**  **if a > b:**  **return a**  **return b** |
| import | được dùng để nhập các mô đun co các hàm và biến được sử dụng trong chương trình hiện tại. | **import math**  **print(math.pi)** |